Lean 4 Cheatsheet

In the following tables, *name* always refers to a name already known to Lean while *new_name* is a new name provided by the user; *expr* means an expression, for example the name of an object in the context, an arithmetic expression that is a function of such objects, a hypothesis in the context, or a lemma applied to any of these; *proposition* is an expression of the type `Prop` (e.g. `0 < x`) When one of these words appears twice in the same cell, the appearances do not designate the same name or the same expression.

Logical symbol	Appears in goal	Appears in hypothesis
$\exists$ (there exists)	`use` *expr*	`obtain` ⟨*new_name* , *new_name*⟩ `:=` *expr*
$\forall$ (for all)	`intro` *new_name*	`apply` *expr* or `specialize` *name expr*
$\neg$ (not)	`intro` *new_name*	`apply` *expr* or `specialize` *name expr*
$\to$ (implies)	`intro` *new_name*	`apply` *expr* or `specialize` *name expr*
$\leftrightarrow$ (if and only if)	`constructor`	`rw [`*expr* `]` or `rw [← ` *expr* `]`
$\wedge$ (and)	`constructor`	`obtain` ⟨*new_name* , *new_name*⟩ `:=` *expr*

$\vee$ (or)	`left` or `right`	`cases` *expr* `with`
		`| inl` *new_name* `=> ...`
		`| inr` *new_name* `=> ...`

In the left-hand column of the following table, the parts in parentheses are optional. The effect of these parts is also in parentheses in the right-hand column.

Tactic	Effect
`exact` *expr*	the goal is satisfied by *expr*
`refine` *expr*	similar to `exact` but allows to leave any number of `?_` in the *expr* to denote holes that will be filled later (creates goals)
`convert` *expr*	prove the goal by transforming it to an existing fact *expr* and create goals for propositions used in the transformation that were not proved automatically
`convert_to` *proposition*	transform the goal into the goal *proposition* and create additional goals for propositions used in the transformation that were not proved automatically
`have` *new_name* : *proposition*	introduce a name *new_name* asserting that *proposition* is true; at the same time, create and focus a goal for *proposition*
`unfold` *name* (`at` *hyp*)	unfold the definition *name* in the goal (or in the hypothesis *hyp*)
`rw [` (`←`) *expr* `]` (`at` *hyp*)	in the goal (or in the hypothesis *hyp*), replace (all occurrences of) the left-hand side (or the right-hand side, if `←` is present) of the equality or equivalence *expr* by its other side
`rw [` *expr* `,` *expr* `,` *expr* `]` (`at` *hyp*)	do more rewrites in the given order (any number of `←` possible)
`calc`	start a proof by calculation (uses transitivity)
`by_cases` *new_name* : *proposition*	split the proof into two cases depending on whether *proposition* is true or false, using *new_name* as name for this hypothesis
`exfalso`	apply the rule “False implies anything” a.k.a. “ex falso quodlibet” (replaces the current goal by `False`)
`by_contra` *new_name*	start a proof by contradiction, using *new_name* as name for the hypothesis that is the negation of the goal
`push_neg` (`at` *hyp*)	push negations in the goal (or in the hypothesis *hyp*); e.g. change $\neg\,\forall$ x, *proposition* to $\exists$ x, $\neg$ *proposition*
`linarith`	prove the goal by a linear combination of hypotheses
`ring`	prove the goal by combining the axioms of a (semi)ring
`simp` (`at` *hyp*)	simplify the goal (or the hypothesis *hyp*) using standard equalities
`exact?`	search for a single existing lemma which closes the goal, also using local hypotheses
`apply?`	search for lemmas whose conclusion matches the goal; suggest those that may be used with `apply` or `refine`
`aesop`	try to solve the goal using magic